REMARKS

Claims 1-44 are currently pending in the subject application and are presently under consideration. Claims 1, 23, 42 and 44 have been amended as shown on pp. 2-9 of the Reply. Claims 2, 10-11, 18-20 and 43 have been canceled.

Favorable reconsideration of the subject patent application is respectfully requested in view of the comments and amendments herein.

I. Rejection of Claims 42-44 Under 35 U.S.C. §101

In the Final Office Action dated December 17, 2007, claims 42-44 stand rejected under 35 U.S.C. §101 because the claimed invention is directed to non-statutory subject matter. Claims 42 and 44 have been amended herein to clearly illustrate that elements within such claims are stored on a computer-readable storage medium. In particular, claim 42 as amended is directed towards a persistent caching system that facilitates truth on a client, the system ***stored on a computer-readable storage medium and comprising a computer processor for executing the following software components***.... (Support for these amendments can be found on pg. 6, lines 15-27). Accordingly, this claim includes functional descriptive material within a computer processor, thereby rendering it structurally and functionally interrelated to the computer processor and is therefore directed to statutory subject matter. Claim 44 has been similarly amended and claim 43 has been canceled, as such this rejection should be withdrawn with regard to claims 42-44.

II. Rejection of Claims 1-44 Under 35 U.S.C. §103(a)

In the Final Office Action dated December 17, 2007, claims 1-44 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Pardikar *et al* (USPGPUB 2004/0236777) and further in view of Mastors *et al* (USPN 5,826,021). Pardikar *et al.* and Mastors *et al.*, individually or in combination, do not teach or suggest each and every element as set forth in the subject claims.

The claimed subject matter relates to a novel client side caching (CSC) infrastructure which facilitates a seamless operation across connectivity states (*e.g.,* online-offline) between client and remote server. More specifically, a persistent caching architecture is employed to safeguard the user (*e.g.,* client) and/or the client applications across connectivity interruptions

and/or bandwidth changes. This is accomplished in part by caching the desirable file(s) together with the appropriate protocol information to a local data store.

In particular, independent claim 1 recites a remote file system that promotes truth on a client, comprising: ... ***a caching component that selectively caches the one or more file objects to a local cache located on a respective client computer, thereby making it available to the client when disconnected from remote location, wherein the caching component silently pushes file objects added by the client to the remote location and triggers a corresponding directory change notification request when physical share connection state has changed, to facilitate effectively enumerating any affected directory;*** *and a component that resolves conflicts between a client version of the one or more file objects and a remote location version of the one or more file objects such that the client version overrides the remote location version when viewed on the client, wherein the component that resolves conflicts is based at least in part upon user preferences; wherein modifications by the client while disconnected from the remote location are stored to the client's memory and then automatically uploaded to the remote location when the client regains connection to the remote location;* ***wherein transitioning online to the remote location is initiated by the caching component which periodically scans offline paths and then initiates an online transition when a path becomes reachable, once the caching component detects a reachable path, it sends an I/O (input/output) control to a driver to initiate an online transition on the reachable path, all existing handles still remain offline until each handle is backpatched individually, once the connection is online, the driver starts to backpatch the handles for directories first to maintain the consistent view of the directory after transitioning online, such that the user retains a consistent view of the file objects even when the file objects have been modified locally but the changes have not been pushed out; and a synchronization component that background synchronizes namespaces not in conflict between the client and the remote location;*** *and wherein the caching component flushes out stale data from local caches based upon comparison of file signatures and comparison of file properties, wherein the file properties comprise time stamp, file size, and revision count.* The cited references do not expressly or inherently disclose the aforementioned novel aspects of applicants' claimed subject matter as recited in the subject claims.

Pardikar *et al.* discloses a system and method for improved client-side caching that transparently caches suitable network files for offline use. A cache mechanism in a network

redirector transparently intercepts requests to access server files, and if the requested file is locally cached, satisfies the request from the cache when possible. For files existing locally, local and remote timestamps may also be compared to ensure that the local file is current. Otherwise the cache mechanism creates a local cache file and satisfies the request from the server, and also fills in sparse cached files as reads for data in ranges that are missing in the cached file are requested and received from the server. (*See* pg. 1, paragraph [0004]).

In contrast, applicants' claimed subject matter discloses a remote file system that promotes truth on a client. One or more client computers communicate with an online remote location to work on file objects. A caching component caches the file objects to a local cache located on a respective client computer, thereby making it available to the client when disconnected from the remote location. Then, transitioning to online is initiated by the caching component as the result of discovering that a path has become reachable. The caching component periodically scans the paths that are offline. A network arrival event can also trigger the caching component to transition the paths. Once the caching component detects a path can be reachable, it sends an IOCTL (I/O control) to a CSC driver to initiate an online transition on this path. The CSC driver simply resets the state of the directory on the transition list and increases the version number. All the existing handles still remain offline until the handle is backpatched individually.

The caching component completes the pending directory change notification on these handles so that the application can send a new directory enumeration to see the online view. To maintain the consistent view of the directory after transitioning online and before outbound synchronization completes, the caching component can merge the results from the server with the results from the cache, such as add, remove, or modify entries on the enumeration buffer depending on the cache files. The namespace as seen by the user is a union of the namespace as it exists on the server and as it exists in the offline store. This ensures that the applications, and hence, the user retains a consistent view of the files after transitioning online automatically, such as file sizes and time stamps, even when the files have been modified locally but the changes have not been pushed out. (*See* pg. 24, lines 1-26).

Pardikar *et al.* merely discloses creating a new file on the server for each file created offline. Further, files marked as modified files are not combined, but rather the user is provided with choices comprising: keep the offline copy, keep the server copy or save the offline copy

under a new name. (*See* pg. 6, paragraph [0046]). If a requested file is not in the cache, the server copy is retrieved and the server copy and the local copy are compared for any changes. Changes are synchronized and included on a modified file indicator maintained as metadata with the file. (*See* pg. 4, paragraph [0032]).

Applicants' claimed system receives an open request for a file and detects whether the file is in conflict with the server version. If a conflict is detected, the caching component satisfies the request with only the local handle and subsequent file I/O operations are performed on the local cache. This allows deferred synchronization as background after a path is transitioned online. This ensures that the applications, and hence, the user retains a consistent view of the files after transitioning online automatically, such as file sizes and time stamps, even when the files have been modified locally but the changes have not been pushed out. This is opposed to Pardikar *et al.*, in which a user manually chooses whether to keep the offline copy or keep the remote server copy and thus does not retain a consistent view of the files after transitioning online.

Furthermore, Mastors *et al.* does not cure the deficiencies of Pardikar *et al.* Mastors *et al.* was cited by the Examiner for disclosing a caching component that periodically scans offline paths and then initiates an online transition when a path becomes reachable. (*See* Final Office Action dated 12-17-07, pg. 4). However, Mastors *et al.* does not disclose a caching component that detects a reachable path and sends an I/O control to a driver to initiate an online transition. The caching component also triggers a corresponding directory change notification request when physical share connection state has changed, to facilitate effectively enumerating any affected directory. Mastors et *al.* merely utilizes a client to detect that the server is available. When the client detects that the server is available, stored credentials are retrieved and sent to the server for verification.

In contrast, Applicants' claimed system receives an open request for a file and detects whether the file is in conflict with the server version. If a conflict is detected, the caching component satisfies the request with only the local handle and subsequent file I/O operations are performed on the local cache. This allows deferred synchronization as background after a path is transitioned online. This ensures that the applications, and hence, the user retains a consistent view of the files after transitioning online automatically, such as file sizes and time stamps, even when the files have been modified locally but the changes have not been pushed out.

Furthermore, independent claim 23 discloses a persistent caching method that facilitates truth on a client, comprising: *selectively caching one or more file objects from a remote server to at least one local cache located on at least one client computer while online; transitioning to an offline state; modifying by a client, a client-cached file object while offline; viewing a client version of the file if it conflicts with the server version; storing modifications by the client while offline in the client's memory; automatically uploading the modifications to the remote location when the client is back online; initiating an online transition via scanning offline paths until a path becomes reachable,* ***sending an I/O (input/output) control to a driver to initiate an online transition on the reachable path, wherein all existing handles still remain offline until each handle is backpatched individually; and backpatching the handles for directories first to maintain the consistent view of the directory after transitioning online, such that the user retains a consistent view of the file objects even when the file objects have been modified locally but the changes have not been pushed out.***

As stated *supra*, Pardikar *et al.* merely discloses creating a new file on the server for each file created offline. Further, files marked as modified files are not combined, but rather the user is provided with choices comprising: keep the offline copy, keep the server copy or save the offline copy under a new name. If a requested file is not in the cache, the server copy is retrieved and the server copy and the local copy are compared for any changes. Changes are synchronized and included on a modified file indicator maintained as metadata with the file. And, Mastors *et al.* merely utilizes a client to detect that the server is available. When the client detects that the server is available, stored credentials are retrieved and sent to the server for verification. In contrast, applicants' claimed subject matter receives an open request for a file and detects whether the file is in conflict with the server version. If a conflict is detected, the caching component satisfies the request with only the local handle and subsequent file I/O operations are performed on the local cache. This allows deferred synchronization as background after a path is transitioned online. This ensures that the applications, and hence, the user retains a consistent view of the files after transitioning online automatically, such as file sizes and time stamps, even when the files have been modified locally but the changes have not been pushed out.

Accordingly, Pardikar *et al.* and Mastors *et al.* do not expressly or inherently disclose a method, comprising: ... ***sending an I/O (input/output) control to a driver to initiate an online transition on the reachable path, wherein all existing handles still remain offline until each***

handle is backpatched individually; and backpatching the handles for directories first to maintain the consistent view of the directory after transitioning online, such that the user retains a consistent view of the file objects even when the file objects have been modified locally but the changes have not been pushed out.

Furthermore, independent claim 42 discloses a persistent caching system that facilitates truth on a client, the system stored on a computer readable storage medium, comprising: *means for selectively caching one or more file objects from a remote server to at means for least one local cache located on at least one client computer while online; means for transitioning to an offline state; means for modifying a client-cached file object while offline; means for viewing a client version of the file if it conflicts with the remote server version; means for storing modifications by the client while offline, in the client's memory; means for automatically uploading the modifications to the remote location when the client is back online; means for initiating an online transition via scanning offline paths until a path becomes reachable;* ***means for sending an I/O (input/output) control to a driver to initiate an online transition on the reachable path, wherein all existing handles still remain offline until each handle is backpatched individually; and means for backpatching the handles for directories first to maintain the consistent view of the directory after transitioning online, such that the user retains a consistent view of the file objects even when the file objects have been modified locally but the changes have not been pushed out.***

As stated *supra*, Pardikar *et al.* merely discloses creating a new file on the server for each file created offline. If a requested file is not in the cache, the server copy is retrieved and the server copy and the local copy are compared for any changes. Changes are synchronized and included on a modified file indicator maintained as metadata with the file. And, Mastors *et al.* merely utilizes a client to detect that the server is available. In contrast, applicants' claimed subject matter receives an open request for a file and detects whether the file is in conflict with the server version. If a conflict is detected, the caching component satisfies the request with only the local handle and subsequent file I/O operations are performed on the local cache. This allows deferred synchronization as background after a path is transitioned online.

Accordingly, Pardikar *et al.* and Mastors *et al.* do not expressly or inherently disclose a method, comprising: ... ***means for sending an I/O (input/output) control to a driver to initiate an online transition on the reachable path, wherein all existing handles still remain offline***

until each handle is backpatched individually; and means for backpatching the handles for directories first to maintain the consistent view of the directory after transitioning online, such that the user retains a consistent view of the file objects even when the file objects have been modified locally but the changes have not been pushed out.

In view of at least the above, it is readily apparent that cited references fail to expressly or inherently disclose applicants' claimed subject matter as recited in independent claims 1, 23 and 42 (and claims 2-22, 24-41 and 44 which respectively depend there from). Accordingly, it is respectfully requested that these claims be deemed allowable.

CONCLUSION

The present application is believed to be in condition for allowance in view of the above comments and amendments. A prompt action to such end is earnestly solicited.

In the event any fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063 [MSFTP528US].

Should the Examiner believe a telephone interview would be helpful to expedite favorable prosecution, the Examiner is invited to contact applicants' undersigned representative at the telephone number below.

Respectfully submitted,

AMIN, TUROCY & CALVIN, LLP

/Himanshu S. Amin/
Himanshu S. Amin
Reg. No. 40,894

AMIN, TUROCY & CALVIN, LLP
24TH Floor, National City Center
1900 E. 9TH Street
Cleveland, Ohio 44114
Telephone (216) 696-8730
Facsimile (216) 696-8731